

1\19

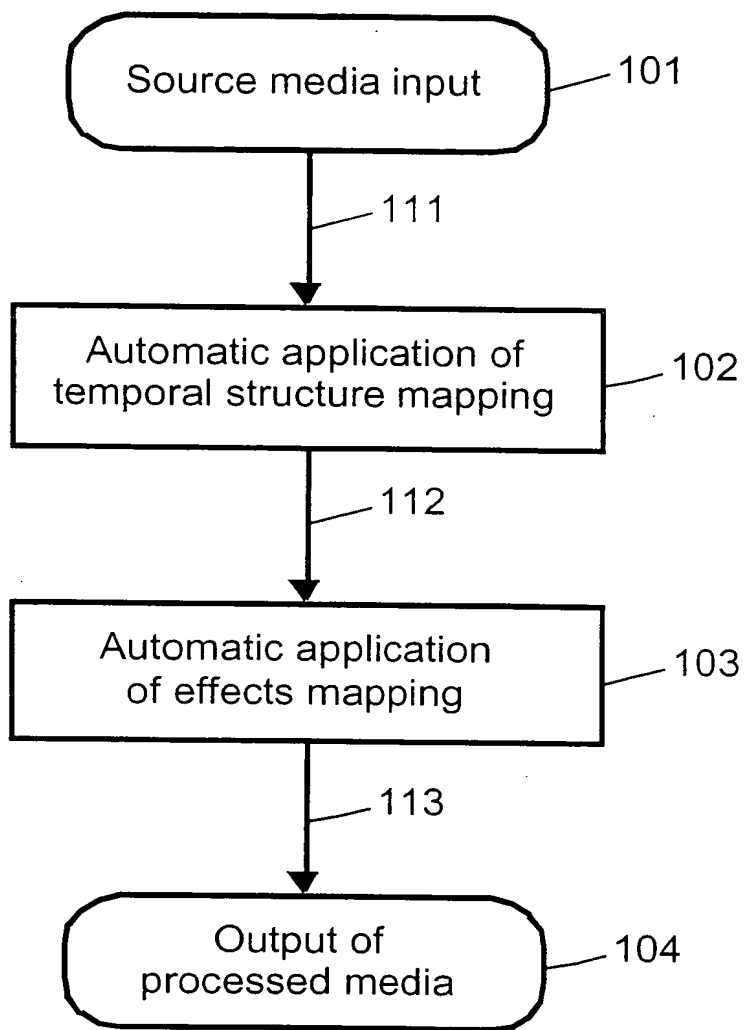
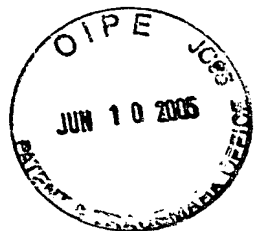


Fig. 1

2\19

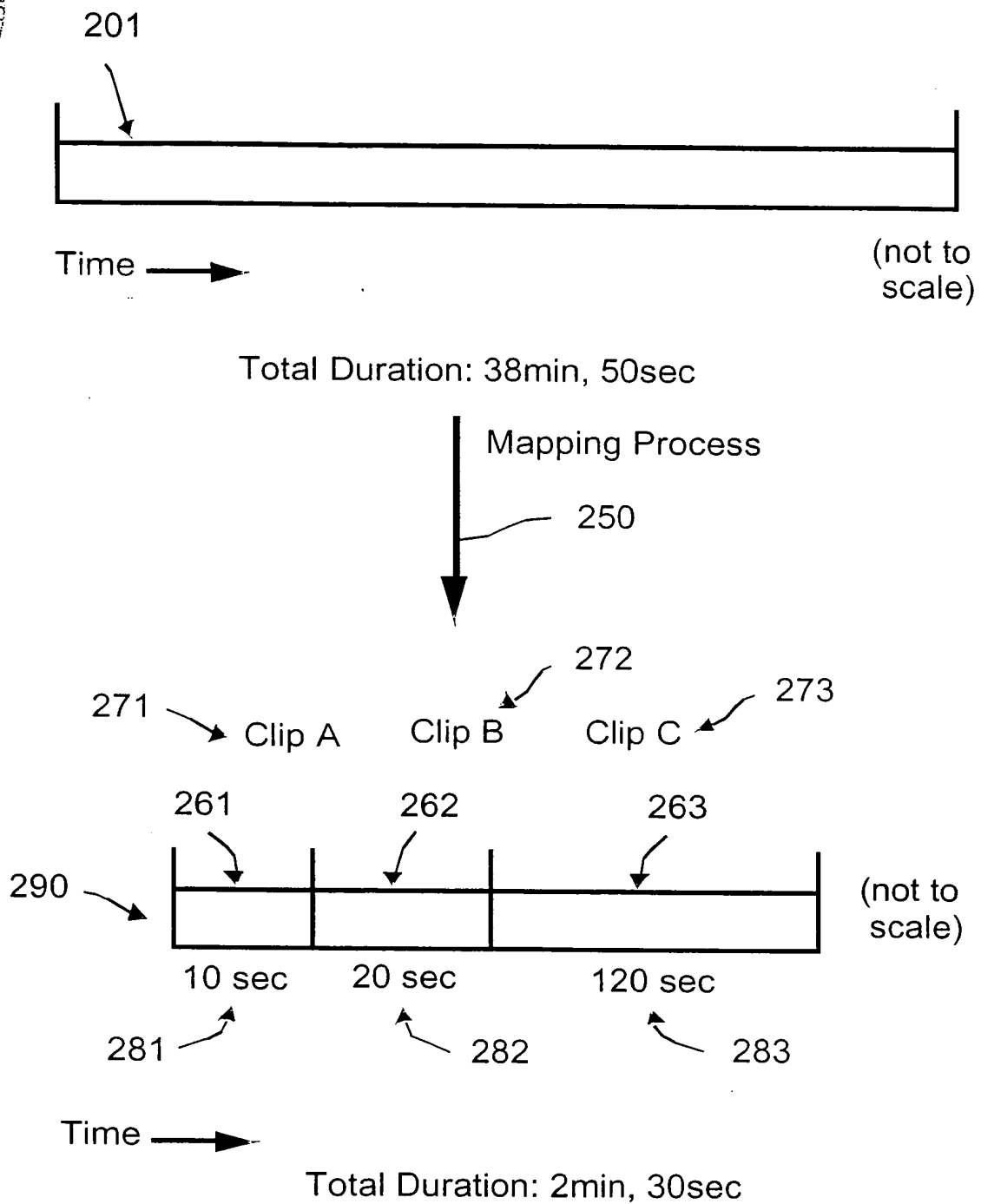


Fig. 2

3\19

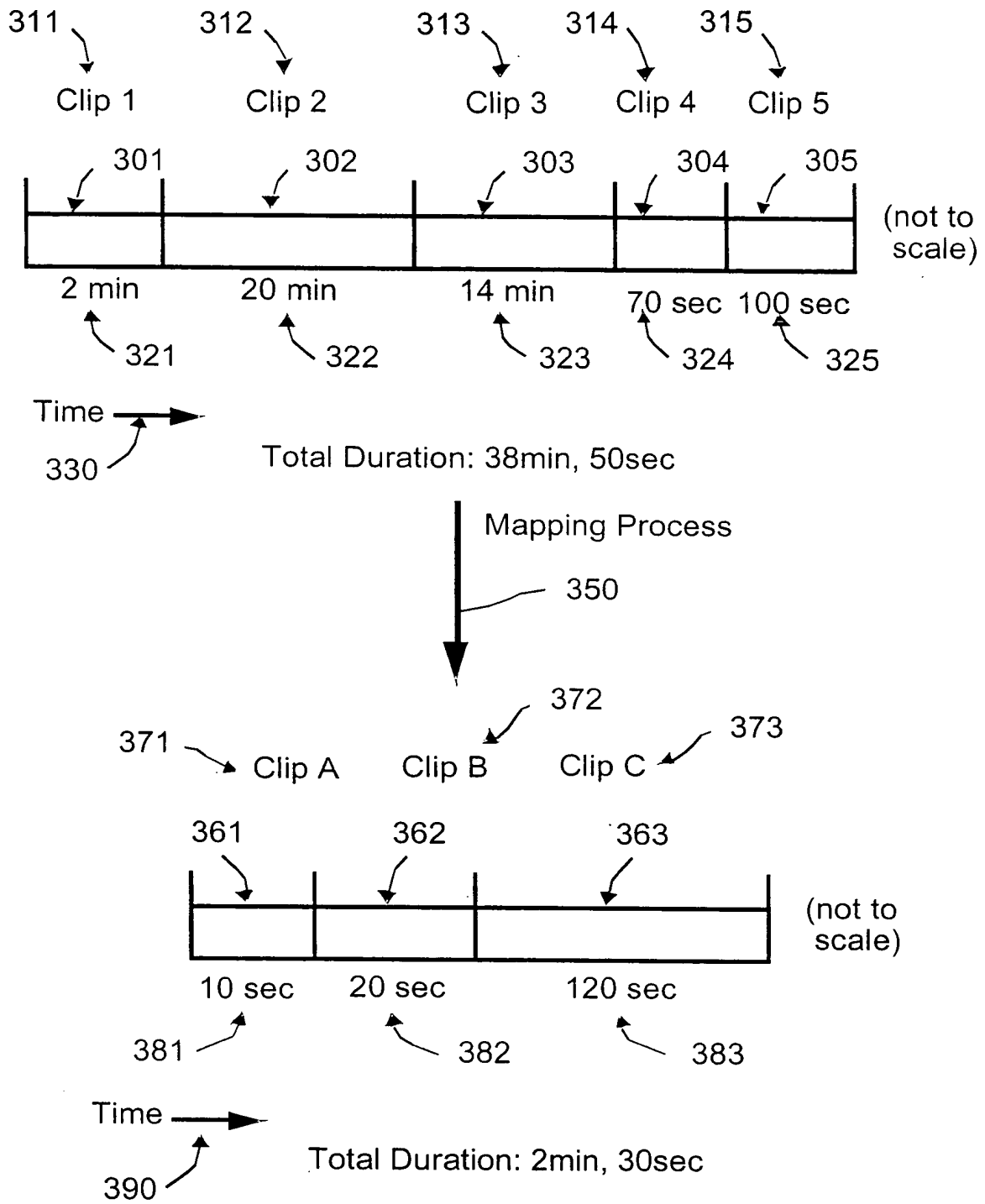


Fig. 3

4\19

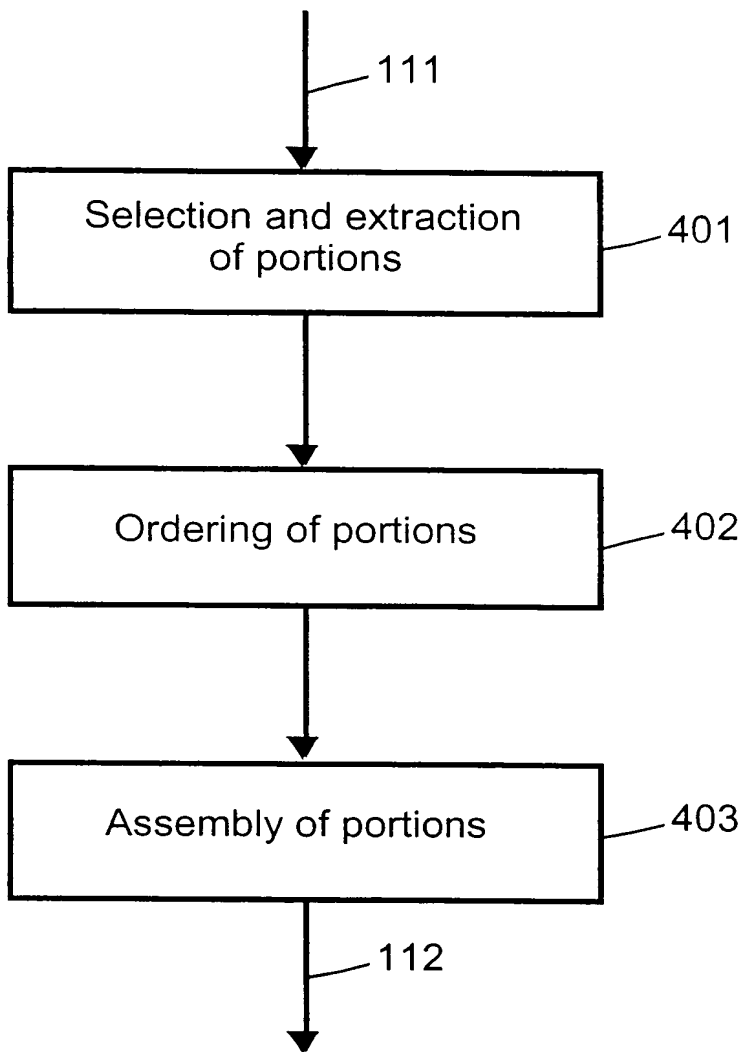


Fig. 4

5\19

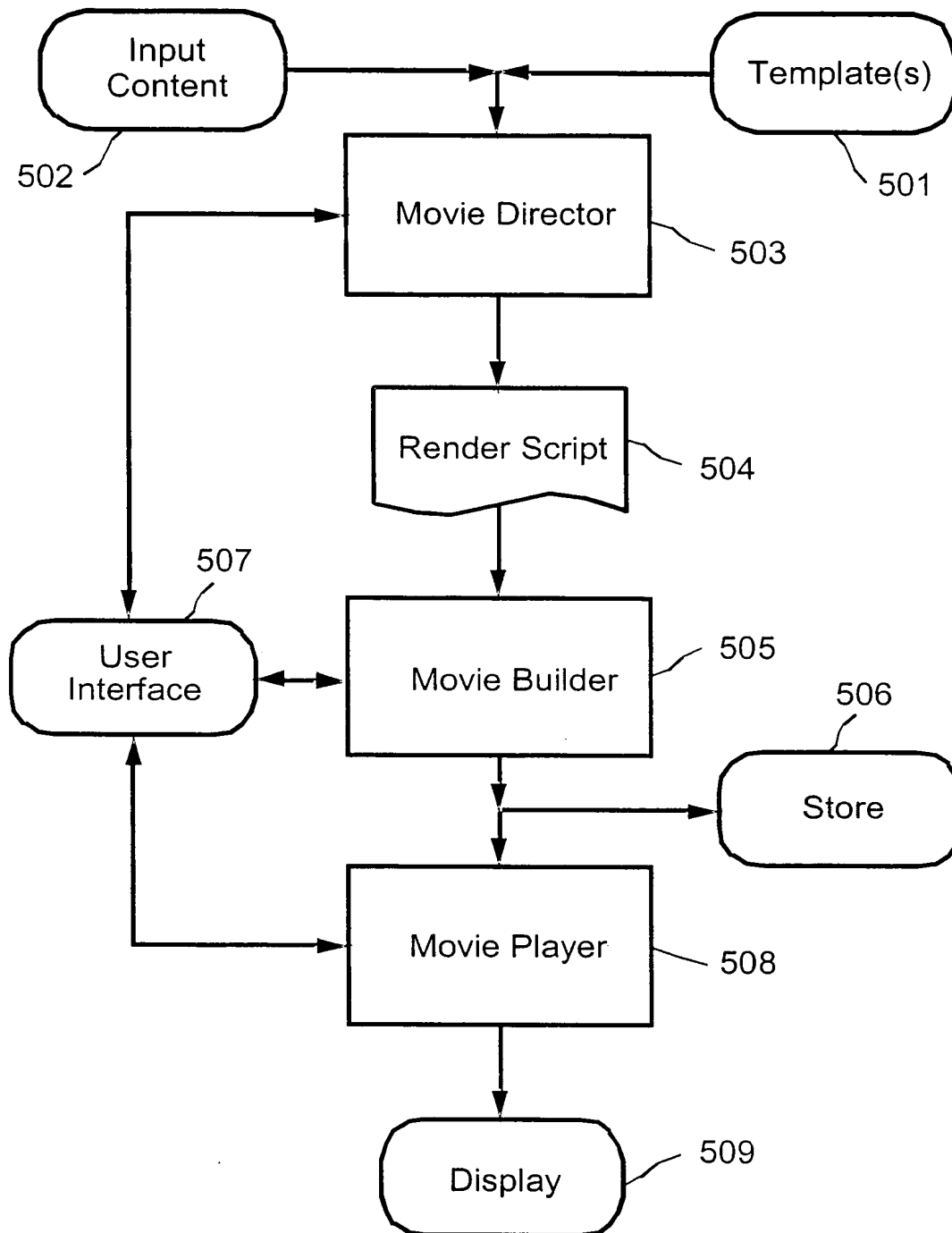


Fig. 5





6\19

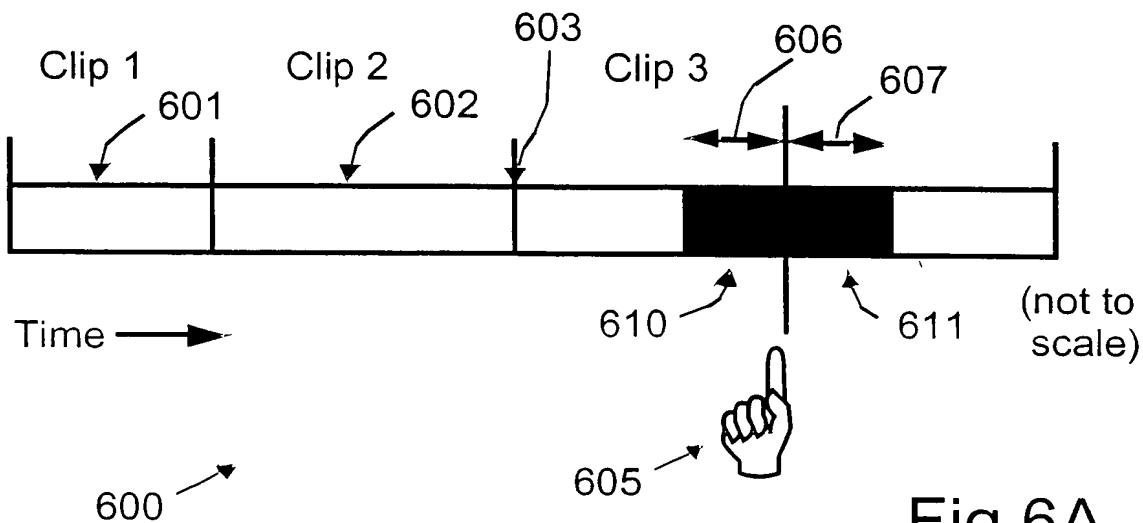


Fig 6A

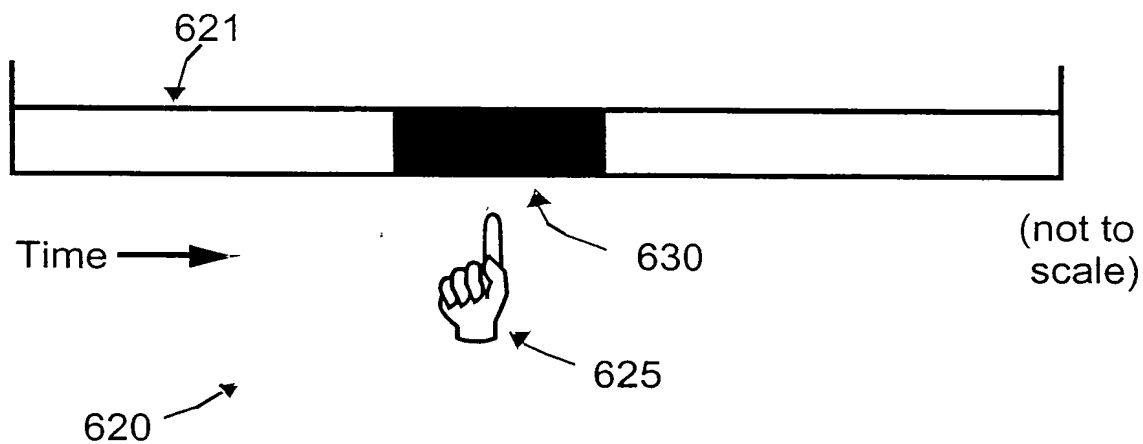
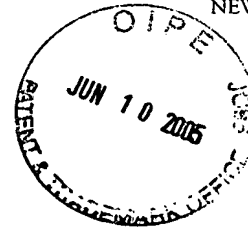


Fig. 6B

7\19



Selection and Extraction Method Examples		
Selected Portion Start	Extracted Duration	Relationships between Selections
Within whole content.	Random.	Random, chronological, without overlap.
Within clip.	Less than or equal to clip length.	Random, chronology ignored, overlap ignored.
Within a group of clips.	Spanning one or more clips recorded within the same day.	From separate clips.
Heuristically obtained, eg. assume zones of interest in recorded content occur primarily near clip startpoints.	Heuristically obtained, eg. related to human attention span.	From a group of clips recorded within the same day.
Multi-pass (repetitious)	Limited so as to limit total output duration (eg. based on heuristics).	From all clips within whole content.
	Short durations followed by longer durations (eg. applied to multipass selection)	Repetitious, for instance, to lengthen output content duration with respect input content duration.

Fig. 8

8\19



Ordering Method Examples
Sequential or chronological
Random
Reverse-chronological
Flashback (later chronologies displayed or duplicated early in the order)
Montage (later chronologies displayed in brief early in the order)
Cutaway (two related or consecutive portions separated by an unrelated or distant portion)
Alternate

Fig. 9

Assembly Method Examples
Cut (butt-edit)
Short Dissolve
Long Dissolve
Fast Wipe
Slow Wipe
Graphic

Fig. 10

9\19



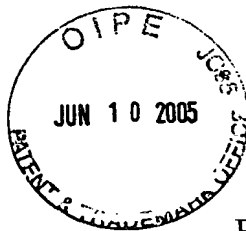
Effects Mapping Examples
Addition of Sound effect
Removal of chrominance
Addition of artificial scratches and dust
Composition or overlay of sprites, animation, graphics
Addition of Music
Luminance or chrominance keying or matteing
Dissolve or mixing of other content

Fig. 11



Silent Movie Template Components Example	
Component	Purpose
Four well-separated random video selections from input content.	Selection of sufficiently differing activities or incidents from the input content to create surprise or reduce boredom.
Extract limited duration clips for each selection, each preferably less than 2 minutes in duration.	Limit clip duration to the effective viewer attention span and avoid boredom.
Filter clips to remove all chrominance information.	Replicate "black and white" characteristic of Silent Movie genre.
Remove original audio information.	Replicate silent characteristic of Silent Movie genre.
Add piano soundtrack.	Replicate characteristic of Silent Movie genre.
Insert dialogue mattes at clip boundaries.	Replicate characteristic of Silent Movie genre.
Apply scratch and dust filter.	Replicate characteristic of Silent Movie genre.
Cut in titles, dialogue mattes and clips.	Replicate hard-cut characteristic of Silent Movie genre.
Insert fade-in from black to title dialogue matte.	Include title in characteristic style of Silent Movie genre.
Insert fade-out to black from end-title dialogue matte.	Include end-title in characteristic style of Silent Movie genre.
Insert film projector sprocket hole sound over title.	Replicate projector sound-effect characteristic of Silent Movie genre.

Fig. 12



11\19

Fig. 13

Example Associations between Editing & Effect Techniques and Template Type				
	Romance Montage	Action Montage	Continuity Template	Silent Movie
Transitions				
Fade				
Fade out	✓		✓	✓
Fade in	✓		✓	✓
Dissolve	✓			
Cross-fades	✓		✓	
Wipe		✓		
Quick/Whip		✓		
Audio	✓		✓	
Sound Types				
Actual Sound	✓	✓	✓	
Sound effects	✓	✓	✓	✓
Atmos sound	✓	✓	✓	✓
Voice over	✓	✓	✓	
Cuts				
Cross cut	✓	✓		
Continuity cut	✓	✓	✓	✓
Compilation cuts		✓		
Split editing	✓		✓	
Parallel cutting				
Classical cutting	✓		✓	✓
Editing effects				
Cutaways	✓	✓	✓	
Insert	✓	✓	✓	
Subliminal cuts				
Flashbacks		✓	✓	
Freeze-frames	✓	✓		
Frequency	✓	✓		
Duration				
Montages	✓	✓		
Rhythm	✓	✓	✓	
Reverse shot	✓	✓	✓	
Shot length				
Same length			✓	
Slow cutting	✓			
Fast cutting	✓	✓		
Cut to beat/music	✓	✓		



12\19

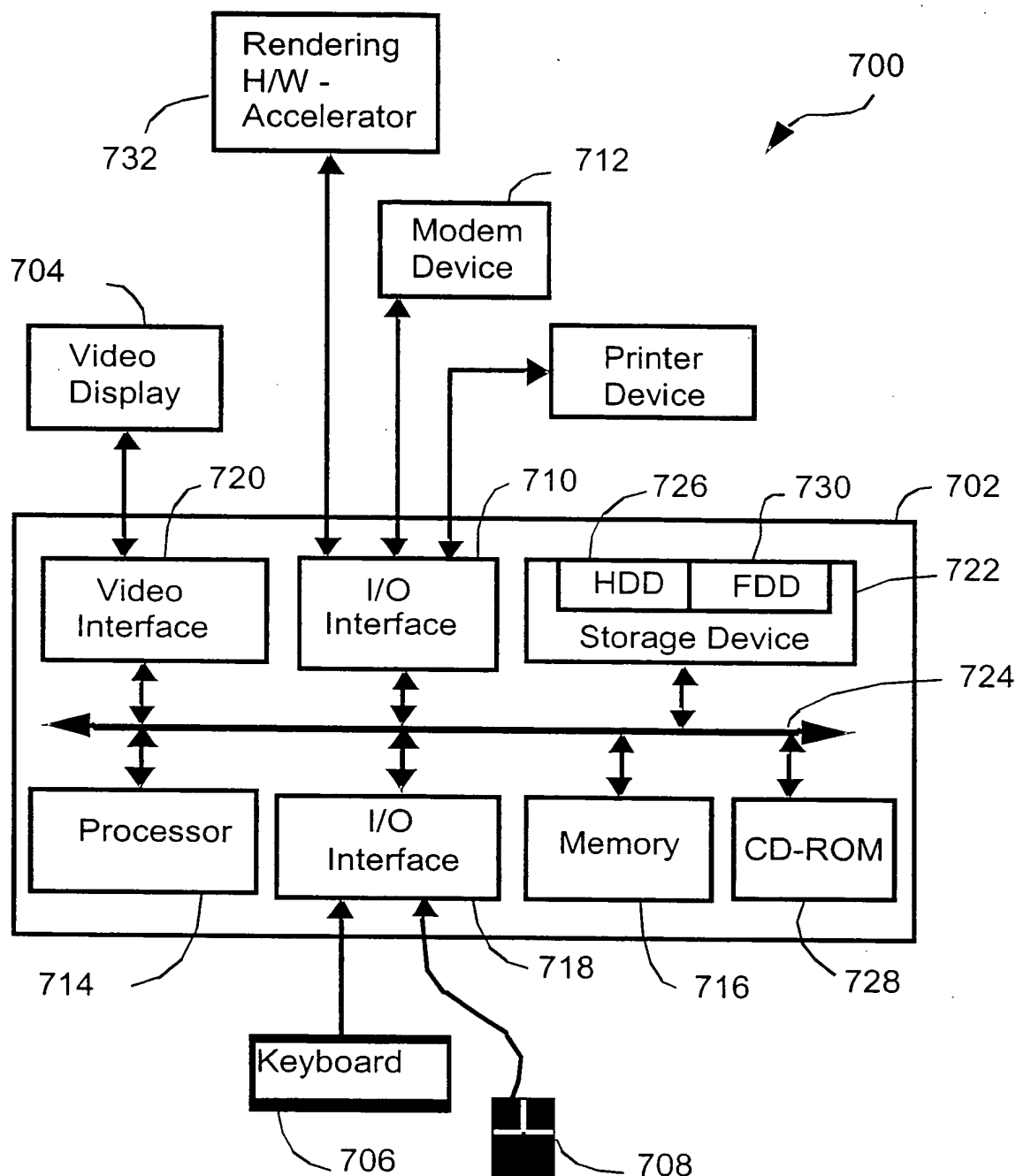


FIG. 7



13\19

Fig. 14A

Movie Director Example Implementation (Pseudo-code)

```
main()
begin
    create rule list
    create parameter list
    create item list
    create rule syntax table

    get template file name
    load(template_file_name)

    get render script file name
    create render script file

    get input content file names
    create content list
    contentparse(content_list, input_content_file_names,...)

    ruleparse(installed_rules, content_list, render_script_file)
    save render_script_file
    close render_script_file
    exit
end

load(template_file_name)
begin
    while(not end of template_file)
    begin
        get next item
        if (item_type == reference)
            resolve(item)
        else if (item_type == rule)
            install rule_name
        else if (item_type == parameter)
            write(parameter_list, parameter_name)
        else if (item_type == rule_syntax_extension)
            write(rule_syntax_table, rule_syntax_extension)
        else
            write(item_list, item_name)
        end
    end
end

resolve(reference_name)
begin
    if (reference_type == provided_content)
```

14\19

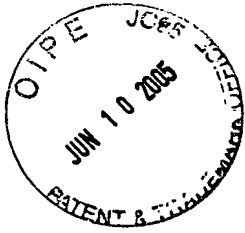


Fig. 14B

```
begin
    get provided content file name
    contentparse(content_list, provided_content_file_name)
end
else
    get referenced item
end
end

contentparse(content_list, content_file_name, ...)
begin
    while(not last content item)
    begin
        if (content_file_name_type == directory)
        begin
            get directory contents
            contentparse(content_list,
                directory_content_file_names,...)
        end
        else
        begin
            get content information
            write(content_list, content_file_name, content_information)
        end
    end
end

ruleparse(rule_list, content_list, render_script_file_name)
begin
    create instruction list
    while (not last rule)
    begin
        get rule
        decode(instructions, operands, rule, content_references,
            parameter_references, item_references)
    end

    get instruction list
    while (not last instruction)
    begin
        execute instruction(operands)
    end
end

decode(instructions, operands, rule, content_references, parameter_references,
    item_references)
begin
    while (not end of rule)
```

15\19



begin

```
get next portion
if (portion_type == instruction)
    begin
        read(portion)
        convert portion according to rule syntax table
        write(instruction_list, instruction)
    end
else
    begin
        read(reference)
        convert portion according to rule syntax table
        write(instruction_list, operand)
    end
end
```

Fig. 14C

end

16\19



Fig. 15A

Movie Builder Example Implementation (Pseudo-code)

```
main()
begin
    get render script file name
    get destination movie file name
    open render script file
    create qt_movie_file
    parse(render_script_file, qt_movie_file)
    close render_script_file
    save qt_movie_file
    close qt_movie_file
    exit
end

parse(script_file_name, qt_movie_file)
begin
    while(not end of script_file)
    begin
        get next script file line
        parse_line(script_file_line, qt_movie_file)
    end
end

parse_line(script_file_line, qt_movie_file)
begin
    get first word of line
    if "/" return
    else if "video" then
        video(script_file_line, qt_movie_file)
    else if "audio" then
        audio(script_file_line, qt_movie_file)
    else if "transition" then
        transition(script_file_line, qt_movie_file)
```


INVENTOR: JAMES ANTHONY
BALNAVES ET AL.
TITLE: METHOD AND APPARATUS FOR
MULTIMEDIA EDITING
Sheet 17 of 19 Appln. no. 09/407,293
Docket No.: 00169.001468

17\19



```
else
    flag error in script file
end

video(script_file_line, qt_movie_file)
begin
    parse video paramenters
    add video to qt_movie_file using QT API
end

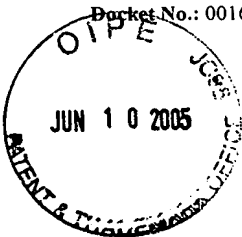
audio(script_file_line, qt_movie_file)
begin
    parse audio paramenters
    add audio to qt_movie_file using QT API
end

transition(script_file_line, qt_movie_file)
begin
    parse transition paramenters
    add transition to qt_movie_file using QT API
end
```

Fig. 15B

INVENTOR: JAMES ANTHONY
BALNAVES ET AL.
TITLE: METHOD AND APPARATUS FOR
MULTIMEDIA EDITING
Sheet 18 of 19 Appln. no. 09/407,293
Docket No.: 00169.001468

18\19



Template Example Implementation (Pseudo-code)

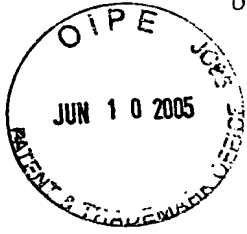
```
//Action Template
cut_order = chronological           Fast-paced, quick cutting, fast beat.
structure = 10s, 4s, ...           //chronology not strictly enforced
intraclip_cutting = 2              //repetitive temporal structure
intraclip_spacing = 2s             //one long clip may contribute 2 elements
avoid_cutting = 1s, -1s           //do not use first/last second of clip
cut_method = random, clip
play_order = forward
structure_transition = 3, 4, crossfade //3-4 frame crossfade
beat_synchronise = true            //sync video clip lengths to music beat
back_track = action                //specify backing music characteristics
audio_action = mute_all           //remove all original audio
title = action_title
end_title = action_end_title

//function definition

length check_fit(content-length, structure, intraclip_spacing, intraclip_cutting,
                  avoid_cutting)
begin
    length = content_length - avoid_cutting[0] + avoid_cutting[1] - structure[0]
    x = intraclip_cutting
    while (x > 1)
        begin
            x = x - 1
            length = length - structure[x] - intraclip_spacing[x]
        end
    return length
end
```

Fig. 16A

19\19



```
main()          //start
begin
    trim (title, beat_synchronise, structure_transition)
    assemble_edit (output, title, play_order, structure_transition, audio_action,
                    back_track)
    while not (completed content list)
    begin
        get next content (cut_order)
        excess = check_fit (content_length, structure, intracclip_cutting,
                             intracclip_spacing, avoid_cutting)
        if (excess > 0)
        begin
            y = 0
            cut_start = cut_method(excess)
            cut_end = 0
            while (y < intracclip_cutting)
            begin
                cut_end = cut (avoid_cutting + cut_start + cut_end,
                               structure[y])
                y = y + 1
                cut_start = excess - cut_start
            end
        end
        trim (current_clips, beat_synchronise, structure_transition)
        assemble_edit (output, current_clips, play_order, structure_transition,
                        audio_action, back_track)
    end
    trim (end_title, beat_synchronise, structure_transition)
    assemble_edit (output, end_title, play_order, structure_transition, audio_action,
                    back_track)
end //finish
```

Fig. 16B